CRT-2000-05

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

PERTURBATION HEURISTICS FOR THE PICKUP AND DELIVERY TRAVELING SALESMAN PROBLEM

by

Jacques Renaud
Fayez F. Boctor
Gilbert Laporte

January 2000

CRT-2000-05

*PERTURBATION HEURISTICS FOR THE PICKUP AND DELIVERY TRAVELING SALESMAN PROBLEM**

by

Jacques Renaud[1,2]
Fayez F. Boctor[2,3]
Gilbert Laporte[4]

* *This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).*

[1] Télé-Université, 2600 boulevard Laurier, Tour de la Cité, 7e étage, C.P. 10700, Sainte-Foy, Canada G1V 4V9

[2] Centre de recherche sur les technologies de l'organisation réseau, Université Laval, Québec Canada G1K 7P4

[3] Faculté des sciences de l'administration, Université Laval, Québec, Canada G1K 7P4

[4] École des Hautes Études Commerciales, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7 and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2000

ABSTRACT

This article describes and compares seven perturbation heuristics for the *Pickup and Delivery Traveling Salesman Problem* (PDTSP). In this problem, a shortest Hamiltonian cycle is sought throught a depot and several pickup and delivery pairs. Perturbation heuristics are diversification schemes which help a local search process move away from a local optimum. Three such schemes have been implemented and compared: *Instance Perturbation*, *Algorithmic Perturbation*, and *Solution Perturbation*. Computational results on PDTSP instances indicate that the latter scheme yields the best results. On instances for which the optimum is known, it consistently produces optimal or near-optimal solutions.

Key words: pickup and delivery traveling salesman problem, perturbation heuristics.

RÉSUMÉ

Dans cet article, on décrit et compare sept heuristiques de perturbation pour le *problème du voyageur de commerce avec collectes et livraisons* (PVCCL). Ce problème consiste à déterminer un cycle hamiltonien de longueur minimale partant du dépôt et visitant plusieurs paires de collecte et livraison. Les heuristiques de perturbation sont des techiques de diversification qui aident un processus de recherche local à s'extirper d'un optimum local. On a développé et comparé trois schémas de perturbation: la *perturbation de données*, la *perturbation d'algorithme* et la *perturbation de solution*. Des résultats numériques obtenus sur le PVCCL indiquent que le dernier type de perturbation produit les meilleurs résultats. Sur des problèmes pour lesquels l'optimum est connu, il permet d'identifier de façon régulière des solutions optimales ou quasi-optimales.

Mots clefs: problème du voyageur de commerce avec collectes et livraisons, heuristiques de perturbation.

1. Introduction

The purpose of this article is to describe and compare several perturbation heuristics for the *Pickup and Delivery Traveling Salesman Problem* (PDTSP) defined as follows. Let $G = (V, E)$ be an undirected graph where $V = \{v_1, ..., v_n\}$ is the vertex set, n is odd, v_1 represents a depot, and $E = \{(v_i, v_j) : i < j, v_i, v_j \in V\}$ is the edge set. The set $V \setminus \{v_1\}$ is partitioned into $\{P, D\}$ where P is a set of *pickup customers*, D is a set of *delivery customers*, and $|P| = |D| = (n-1)/2$. These two sets are twinned in the sense that to each pickup customer $v_i \in P$ corresponds exactly one delivery customer $d(v_i) \in D$, and to each delivery customer corresponds exactly one pickup customer. A distance matrix $C = (c_{ij})$ is defined on E. In what follows, c_{ij} must be interpreted as c_{ji} whenever $i > j$. The PDTSP consists of determining a shortest Hamiltonian cycle on G starting and ending at the depot, and such that each pickup customer is visited before its associated delivery customer. The PDTSP is NP-hard since it reduces to the *Traveling Salesman Problem* (TSP) whenever each pickup customer v_i coincides with its associated delivery customer $d(v_i)$.

The PDTSP is related to, but different from, the *Dial-a-Ride Problem* (DARP) in the following way: in the DARP there may be several capacitated vehicles instead of only one, time windows are often present, and the objective function may be different. For example, one may wish to minimize the total distance traveled between each pickup point and its associated delivery point. For references on the DARP, see, e.g., Psaraftis (1980, 1983, 1986), Sexton and Bodin (1985a, 1985b), Sexton and Choi (1986), Desrosiers, Dumas and Soumis (1986), Jaw et al. (1986), Dumas, Desrosiers and Soumis (1991), Van der Bruggen, Lenstra and Shuur (1993), Savelsbergh and Sol (1995). Two other problems related to the PDTSP are the *Vehicle Routing Problem with Backhauls* (Casco, Golden and Wasil, 1988, Goetschalckx and Jacobs-Blecha, 1989, Mingozzi, Giorgi and Baldacci, 1999), and the *TSP with Backhauls* (Gendreau, Hertz and Laporte, 1996, 1997). In both problems, all pickup customers must be visited before delivery customers but there is no relationship between these two sets.

Relatively few algorithms have been proposed for the PDTSP. Kalantari, Hill and Arora (1985) have described an exact branch-and-bound procedure for this problem and have applied to instances involving no more than 40 vertices. Savelsbergh (1990) and Healy and Moll (1995) have proposed several heuristic improvement schemes based on edge interchanges. More recently, Renaud, Boctor and Ouenniche (2000) have implemented an edge interchange scheme as well as a vertex deletion and reinsertion method. As far as we are aware, no heuristic method has been assessed by making comparisons with optimal solutions.

Our aim is to develop several types of pertubation heuristics for the PDTSP. These can be viewed as *diversification* schemes that can help an improvement algorithm to escape from a local optimum, similar to what is sometimes done in tabu search (Glover and Laguna, 1997). A first type of perturbation heuristic, called *Instance Perturbation* (IP) was introduced by Storer, Wu and Vaccari (1992), Charon and Hudry (1993) and later implemented by Codenetti, Manzini, Margara and Resta (1996) in the context of the TSP. In IP, when a local optimum is reached, the instance data are marginally perturbed, an improvement algorithm is then applied to the modified data, and the local optimum of the perturbed instance is translated back into the original data. The same process can be applied iteratively.

A second type of perturbation is called *Algorithmic Perturbation* (AP). This concept can be applied to a construction or to an improvement heuristic. In the first case, the criterion used to generate a feasible solution can be modified from one iteration to the next. In the second case, it is useful to view an improvement algorithm as a process that iteratively moves from a solution to another solution in its neighbourhood. For example, the neighbour of a TSP solution can be any solution reachable by removing and reintroducing k edges. To avoid local optima, one can occasionally modify the rule that governs the definition of the neighbourhood, e.g., by going from 3–opt to 4–opt. Examples of AP are multistart methods (Eglese, 1990) and variable neighbourhood search (Boctor 1993, Mladenovic and Hansen 1997).

A third type of perturbation is *Solution Perturbation* (SP). Here, a local optimum is modified and the improvement procedure is reapplied to the perturbed solution. A common example of SP is the mutation process in genetic algorithms (Dowsland, 1996). For an interesting application of SP to the *Quadratic Assignment Problem*, see Fleurent and Ferland (1996).

The remainder of this article is organized as follows. In Section 2, we describe several implementations of IP, AP and SP to the PDTSP. Extensive computational results are presented in Section 3, followed by the conclusion in Section 4.

2. Perturbation heuristics

We have developed a total of seven perturbation heuristics for the PDTSP. All proceed according to the following general framework:

Step 1. (Initialization). Construct a feasible PDTSP solution and improve it by means of 4–opt**, an adaptation of the 4–opt* heuristic developed by Renaud, Boctor and Laporte (1996). Basically, 4–opt* uses eight of the forty-eight potential 4–opt (Lin, 1965) moves while ensuring that at least one added edge will be shorter than a removed edge. Our adaptation of 4–opt* is such that $v_i \in P$ always appears before $d(v_i)$ when the tour is initialized at v_1.

Step 2. (Perturbation). Apply a perturbation scheme (either IP, AP or SP).

Step 3. (Postoptimization). Apply the 4–opt** heuristic to one or several solutions obtained at the end of Step 2. If a stopping criterion is satisfied, stop. Otherwise go to Step 2. If IP or SP are used, the application of Step 2 at the end of Step 3 is initiated from either the incumbent or from the current solution, at the choice of the user. We have tested these two possibilities.

In all cases, the best known solution is kept in memory and the algorithm terminates when the incumbent has not improved for λ successive applications of Steps 2 and 3, where λ is a user-controlled parameter. We now describe each step of the general framework.

2.1 Initialization

The construction heuristic starts by determining the pair $(v_{i^\bullet}, v_{j^\bullet} = d(v_{i^\bullet}))$ yielding the *longest* tour, and then sequentially inserts the remaining pairs $(v_i, d(v_i))$ yielding at each step the minimum value of a score function.

Step 1. (Initial subtour). Determine the vertex pair $(v_{i^\bullet}, v_{j^\bullet} = d(v_{i^\bullet}))$ yielding $\max_{v_i \in P} \{c_{1i} + c_{ij} + c_{j1}\}$, where $v_j = d(v_i)$. This is the MAX TRIANGLE rule used in I^3 for non-Euclidean instances (see Renaud, Boctor and Laporte, 1996). Set $P := P \setminus \{v_{i^\bullet}\}$.

Step 2. (Vertex insertion). If $P = \varnothing$, go to Step 3. Otherwise determine the vertex pair $(v_{i^\bullet}, v_{j^\bullet} = d(v_{i^\bullet}))$ yielding the best score value. Two cases are possible.

- *Case 1.* $v_{i^\bullet}$ and $v_{j^\bullet}$ are inserted consecutively between vertices v_k and v_ℓ. Let F be the set of all edges of the current subtour. Then

$$\text{SCORE 1} = \min_{v_i \in P, (v_k, v_\ell) \in F} \{\alpha c_{ki} + c_{ij} + (2-\alpha) c_{j\ell} - c_{k\ell}\},$$

  where $v_j = d(v_i)$ and α is a user-controlled parameter $(0 \leq \alpha \leq 2)$.

- *Case 2.* $v_{i^\bullet}$ and $v_{j^\bullet}$ are inserted between vertices v_k, v_ℓ and v_r, v_s, respectively, where (v_r, v_s) appears after (v_k, v_ℓ) on the subtour. Then

$$\text{SCORE 2} = \min_{v_i \in P, (v_k, v_\ell), (v_r, v_s) \in F} \{\alpha(c_{ki} + c_{i\ell} - c_{k\ell}) + (2-\alpha)(c_{rj} + c_{js} - c_{rs})\},$$

  where $v_j = d(v_i)$ and α is a user-controlled parameter $(0 \leq \alpha \leq 2)$.

The vertex pair $(v_{i^\bullet}, d(v_{i^\bullet}))$ yielding $\min\{\text{SCORE 1}, \text{SCORE 2}\}$ is then inserted in its appropriate position in the subtour and $P := P \setminus \{v_{i^\bullet}\}$. Repeat this step.

Step 3 (4–opt**). Attempt to improve the current solution by means of 4–opt**. In our implementation, we use $\alpha = 1.25$.

Several values of α were tested ($\alpha = 0.5, 0.75, 1.00, 1.25$ and 1.50) and the value $\alpha = 1.25$ was retained.

2.2 Perturbation

We have developed two IP schemes (IP1 and IP2), two AP schemes (AP1 and AP2), and three SP schemes (SP1, SP2 and SP3). Their descriptions now follow.

Perturbation scheme IP1

This perturbation scheme applies to planar instances only. It operates with two user-controlled parameters β and γ, where $0 < \beta \leq 1$ and $\gamma > 0$.

Step 1 (Vertex moves). Move each vertex $v_i \in V \setminus \{v_1\}$ with probability β. When vertex v_i is selected for a move, it is relocated within a circle of radius γc_{1i} centered at its current position. Compute the modified distance matrix C', and update the tour length.

Step 2 (4–opt**). Apply 4–opt** to the perturbed instance.

Step 3 (Mapping). Map each vertex of the tour obtained at the end of Step 2 onto its initial position, and update the tour length.

Perturbation scheme IP2

Only Step 1 differs from the previous scheme.

Step 1 (Vertex moves). Consider the current solution $(v_1 \equiv v_{i_1}, v_{i_2}, ..., v_{i_n}, v_1)$. For $t = 2, ..., n$, move vertex v_{i_t} with probability β in the crown centered at $v_{i_{t-1}}$, determined by the two radii $c_{i_{t-1}i_t}$ and $(1+\gamma)c_{i_{t-1}i_t}$.

Perturbation scheme AP1

In this scheme, Step 2 of the general framework is used to generate initial solutions differently (and later postoptimize them in Step 3) in the hope of generating a better local optimum than the initial solution obtained at the end of Step 1. The construction algorithm described in Section 2.1 is perturbed as follows. Steps 1 and 3 are identical. In Step 2, instead of seeking the vertex pair $(v_{i^*}, d(v_{i^*}))$ minimizing $\min\{\text{SCORE 1}, \text{SCORE 2}\}$ with $\alpha = 1$, this choice is now done randomly among all non-inserted vertices $v_i \in P$.

Perturbation scheme AP2

Here the construction algorithm of Section 2.1 is perturbed by using a randomly selected value of α in the interval $[0, 2]$.

Perturbation scheme SP1

This scheme works with two parameters δ and θ. The current solution is perturbed by randomly removing between δ and θ vertices from the current tour and reinserting each of them in a random but feasible position in the tour.

Perturbation scheme SP2

The current solution is again perturbed by randomly removing between δ and θ vertices from the current tour. However, reinsertions are performed by using a least insertion length criterion while maintaining feasibility. The effect of this perturbation rule is likely to be less important than that of SP1 since the insertion criterion may relocate vertices in their original position.

Perturbation scheme SP3

This scheme is inspired from the cross-over operation in genetic algorithms. Two new solutions are created by combining the solution S_1 produced by $SP1$ with the best known solution S_2 different from S_1. A cross-over position u is randomly selected between $\lceil \pi n \rceil$ and $\lceil \sigma n \rceil$, where π and σ are two user-controlled parameters taken as $\pi = 0.3$ and $\sigma = 0.6$ in our tests. To generate a new solution S', the first u vertices $v_1 \equiv v_{i_1}, v_{i_2}, ..., v_{i_u}$ of S_1 are kept in this order, and S' completed by considering in turn the vertices of S_2. Whenever a vertex of S_2 not already present in S' is identified, it is introduced after the vertices of S', in the same sequence in which they appear in S_2. A second solution S'' is also created by keeping this time the first u vertices of S_2 and completing the solution by using in turn the vertices of S_1.

2.3 Postoptimization

The postoptimization heuristic 4–opt** is applied to the solution obtained at the end of the perturbation step (in the case of IP1, IP2, AP1, AP2, SP1 and SP2), or to the two solutions S' and

S'' generated by SP3.

3. Computational results

The algorithms just described were coded in Borland Delphi 3.0 and run on a PC Pentium II, 200 Mhz under Windows 95.

We have tested the perturbation heuristics on two classes of instances. The first class was generated from 36 of the TSPLIB (Reinelt, 1991) Euclidean instances, ranging from 51 to 441 vertices, with distances rounded up or down to the nearest integer. If the number of vertices is even, the last one is dropped. The depot is always the first vertex. Then each pickup-delivery pair was defined as follows: randomly select a pickup vertex and associate to it a delivery vertex according to one of three proximity rules.

- Rule A: select the delivery vertex from among the five closest neighbours (not yet selected) of the pickup vertex.
- Rule B: select the delivery vertex from among the ten closest neigbours (not yet selected) of the pickup vertex.
- Rule C: select the delivery vertex randomly from the set of all unselected vertices.

For the second class of instances, we defined 10 instances of size $n = 101$, and 10 instances of size $n = 201$ as follows. First, n vertices were randomly generated in the $[0, 100]^2$ square, according to a continuous uniform distribution and Euclidean distances, rounded up or down to the nearest integer, were computed between these vertices. The TSP associated with each instance was then solved optimaly using the Padberg and Rinaldi (1991) branch-and-cut algorithm. The first generated vertex was designated as the depot. Then, pickup-delivery pairs were obtained by following the tour. The pickup vertex was taken as the next unselected vertex, and its associated delivery vertex was randomly selected from among the unselected vertices. This procedure ensures that the optimal TSP solution always yields an optimal PDTSP solution.

For the first class of instances, the best known solution was obtained from 136 runs used in the Renaud, Boctor and Ouenniche (2000) experiments, and 168 applications of our perturbation heuristics, run under various rules and with various sets of parameters.

Several independent runs of the IP, AP and SP heuristics were performed on the two classes of instances using several combinations of parameters and variants (applying Step 2 to the incumbent or to the current solution). These combinations are listed in Table 1, together with our recommend selection.

Table 1 Combinations and variants tested for the perturbation heuristics

Heuristics	*List*	*Number of combinations*	*Recommended selection*
IP	IP1, IP2; $\beta = 0.5, 1$; $\gamma = 0.1, 0.2$; $\lambda = 5, 10, 20, 50$; Step 2: incumbent, current.	64	IP1, $\beta = 0.5$; $\gamma = 0.2$; $\lambda = 50$; incumbent. IP2, $\beta = 1.0$; $\gamma = 0.2$; $\lambda = 50$; current.
AP	AP1, AP2; $\lambda = 5, 10, 20, 50$.	8	AP1: $\lambda = 50$. AP2: $\lambda = 50$.
SP	SP1, SP2, SP3; $(\delta, \theta) = (5, 10), (10, 15), (0.05n, 0.10n), (0.10n, 0.15n)$; $\lambda = 5, 10, 20, 50$; Step 2: incumbent, current.	96	SP1: $(\delta, \theta) = (10, 15)$; $\lambda = 50$; incumbent. SP2: $(\delta, \theta) = (0.10n, 0.15n)$; $\lambda = 50$; incumbent. SP3: $(\delta, \theta) = (0.05n, 0.10n)$; $\lambda = 50$; incumbent.

Computational results are presented in Tables 2 to 7 using the recommended selection of parameters and variants. The table headings are defined as follows:

n	:	number of vertices;
Proximity rule	:	rule employed to generate the delivery customers in class 1 instances;
Value	:	solution value produced by the heuristic;
Best	:	best known solution value;
Opt	:	optimal solution value for class 2 instances;
Seconds	:	computing time in seconds;
IP1, IP2, AP1, AP2, SP1, SP2, SP3	:	perturbation heuristic;
RBO	:	Renaud, Boctor and Ouenniche (2000) heuristic.

Table 2 Instance Perturbation, Class 1 Instances

n	Proximity rule	Number of instances	IP1		IP2		RBO
			Value/Best	Seconds	Value/Best	Seconds	Value/Best
50-99	A	11	1.018	36	1.051	26	1.058
	B	11	1.021	34	1.037	30	1.056
	C	11	1.028	28	1.052	30	1.076
100-199	A	14	1.022	186	1.051	104	1.067
	B	14	1.029	150	1.063	110	1.072
	C	14	1.014	167	1.036	87	1.052
200-299	A	5	1.047	727	1.071	260	1.075
	B	5	1.044	803	1.069	292	1.038
	C	5	1.037	615	1.069	473	1.102
300-499	A	6	1.090	1609	1.058	1407	1.089
	B	6	1.066	1149	1.041	1018	1.072
	C	6	1.073	1299	1.049	1140	1.066
All	A	36	1.036	452	1.055	319	1.069
	B	36	1.035	372	1.052	262	1.069
	C	36	1.031	376	1.047	298	1.069
Global average		108	1.034	399	1.052	294	1.069

Table 3 Instance Perturbation, Class 2 Instances

n	Instance number	Opt	IP1			IP2		
			Value	Value/Opt	Time	Value	Value/Opt	Time
101	1	799	799	1.000	32	822	1.029	13
	2	729	729	1.000	27	753	1.033	17
	3	748	748	1.000	13	748	1.000	13
	4	807	807	1.000	18	807	1.000	18
	5	783	783	1.000	15	788	1.006	21
	6	755	776	1.028	17	776	1.028	29
	7	767	767	1.000	19	800	1.043	16
	8	762	762	1.000	21	783	1.028	17
	9	766	766	1.000	22	854	1.115	15
	10	754	754	1.000	14	754	1.000	13
	Average	767.0	769.1	1.003	19	788.5	1.028	17

Table 3 (Continued) Instance Perturbation, Class 2 Instances

201	1	1038	1039	1.001	138	1051	1.013	86
	2	1086	1102	1.018	159	1154	1.063	85
	3	1070	1073	1.003	165	1134	1.060	155
	4	1050	1051	1.001	157	1142	1.088	73
	5	1052	1054	1.002	110	1099	1.045	73
	6	1059	1064	1.005	87	1103	1.042	62
	7	1036	1036	1.000	80	1120	1.081	93
	8	1079	1079	1.000	110	1140	1.057	84
	9	1050	1057	1.007	130	1065	1.014	69
	10	1085	1085	1.000	101	1186	1.093	68
	Average	1060.5	1064.0	1.003	123	1119.4	1.055	84
Global average		913.8	916.6	1.003	71	954.0	1.042	51

Table 4 Algorithm Perturbation, Class 1 Instances

n	Proximity rule	Number of instances	AP1		AP2		RBO
			Value/Best	Seconds	Value/Best	Seconds	Value/Best
50-99	A	11	1.040	38	1.038	148	1.058
	B	11	1.051	34	1.043	173	1.056
	C	11	1.052	45	1.039	159	1.076
100-199	A	14	1.062	186	1.055	1475	1.067
	B	14	1.069	172	1.048	1498	1.072
	C	14	1.049	131	1.034	1358	1.052
200-299	A	5	1.084	469	1.064	9873	1.075
	B	5	1.101	535	1.073	8030	1.083
	C	5	1.119	503	1.056	8965	1.102
300-499	A	6	1.122	1854	1.067	79276	1.089
	B	6	1.085	2052	1.066	94869	1.072
	C	6	1.110	1924	1.075	75035	1.066
All	A	36	1.069	457	1.053	15169	1.069
	B	36	1.070	494	1.053	17563	1.069
	C	36	1.070	456	1.045	14329	1.069
Global average		108	1.070	469	1.050	15687	1.069

Table 5 Algorithm Perturbation, Class 2 Instances

n	Instance number	Opt	AP1			AP2		
			Value	Value/Opt	Time	Value	Value/Opt	Time
101	1	799	805	1.008	61	804	1.006	181
	2	729	731	1.003	45	750	1.029	214
	3	748	752	1.005	36	748	1.000	187
	4	807	825	1.022	49	847	1.050	207
	5	783	788	1.006	35	798	1.019	164
	6	755	756	1.001	86	799	1.058	182
	7	767	775	1.010	34	805	1.050	158
	8	762	773	1.014	45	773	1.014	178
	9	766	766	1.000	52	823	1.074	178
	10	754	759	1.007	84	756	1.003	164
	Average	767.0	773.0	1.008	52	790.3	1.030	181
201	1	1038	1127	1.086	205	1065	1.026	3182
	2	1086	1204	1.109	202	1189	1.095	3605
	3	1070	1164	1.088	180	1123	1.050	2889
	4	1050	1162	1.107	188	1083	1.031	2876
	5	1052	1165	1.107	174	1148	1.091	2891
	6	1059	1131	1.068	180	1087	1.026	2823
	7	1036	1146	1.106	185	1077	1.040	2973
	8	1079	1160	1.075	181	1105	1.024	3085
	9	1050	1082	1.031	161	1082	1.031	2456
	10	1085	1217	1.122	171	1157	1.066	3306
	Average	1060.5	1155.8	1.090	182	1111.6	1.048	3008
Global average		913.8	964.4	1.049	117	950.9	1.039	1595

Table 6 Solution Perturbation, Class 1 Instances

n	Proximity rule	Number of instances	SP1		SP2		SP3		RBO
			Value/Best	Seconds	Value/Best	Seconds	Value/Best	Seconds	Value/Best
50-99	A	11	1.016	30	1.060	7	1.026	37	1.058
	B	11	1.024	22	1.054	11	1.023	35	1.056
	C	11	1.024	19	1.068	11	1.022	27	1.076
100-199	A	14	1.036	112	1.070	30	1.026	217	1.067
	B	14	1.030	102	1.071	41	1.017	206	1.072
	C	14	1.022	76	1.047	36	1.019	168	1.052

Table 6 (Continued) Solution Perturbation, Class 1 Instances

200-299	A	5	1.066	328	1.074	129	1.036	1275	1.075
	B	5	1.044	292	1.078	201	1.044	891	1.083
	C	5	1.040	423	1.084	218	1.043	788	1.102
300-499	A	6	1.069	932	1.089	520	1.053	2582	1.089
	B	6	1.046	894	1.054	342	1.052	2131	1.072
	C	6	1.048	980	1.068	568	1.059	2114	1.066
All	A	36	1.040	254	1.070	117	1.032	703	1.069
	B	36	1.032	235	1.064	136	1.029	570	1.069
	C	36	1.024	258	1.062	142	1.030	536	1.069
Global average		108	1.034	248	1.066	133	1.030	603	1.069

Table 7 Solution Perturbation, Class 2 Instances

n	Instance number	Opt	SP1			SP2			SP3		
			Value	Value/Opt	Time	Value	Value/Opt	Time	Value	Value/Opt	Time
101	1	799	799	1.000	19	820	1.026	7	799	1.000	24
	2	729	729	1.000	11	741	1.017	12	729	1.000	22
	3	748	748	1.000	9	748	1.000	4	748	1.000	7
	4	807	807	1.000	12	874	1.083	6	807	1.000	19
	5	783	787	1.005	9	812	1.037	4	783	1.000	15
	6	755	755	1.000	13	851	1.127	5	755	1.000	9
	7	767	767	1.000	21	804	1.048	6	767	1.000	15
	8	762	762	1.000	12	791	1.038	6	762	1.000	15
	9	766	766	1.000	14	854	1.115	5	766	1.000	17
	10	754	754	1.000	9	763	1.012	4	756	1.003	9
	Average	767.0	767.4	1.001	12	805.8	1.050	5	767.2	1.001	15
201	1	1038	1039	1.001	97	1043	1.005	63	1039	1.001	106
	2	1086	1086	1.000	129	1180	1.087	40	1086	1.000	115
	3	1070	1072	1.002	99	1147	1.072	55	1070	1.000	178
	4	1050	1056	1.006	47	1071	1.020	58	1057	1.007	81
	5	1052	1054	1.002	70	1095	1.041	43	1052	1.000	155
	6	1059	1059	1.000	47	1085	1.025	40	1059	1.000	95
	7	1036	1036	1.000	50	1146	1.106	22	1036	1.000	89
	8	1079	1080	1.001	73	1151	1.067	23	1079	1.000	102
	9	1050	1056	1.006	60	1077	1.026	18	1057	1.007	100
	10	1085	1103	1.017	76	1189	1.096	30	1085	1.000	176
	Average	1060.5	1064.1	1.003	74	1118.4	1.054	39	1062.2	1.001	119
Global average		913.8	915.8	1.002	43	962.1	1.052	22	914.6	1.001	66

Table 8 Comparison of the seven perturbation schemes

Heuristics	Class 1 instances		Class 2 instances	
	Value/Best	Seconds	Value/Opt	Seconds
IP1	1.034	399	1.003	71
IP2	1.052	294	1.042	51
AP1	1.070	469	1.049	117
AP2	1.050	15687	1.039	1595
SP1	1.034	248	1.002	43
SP2	1.066	133	1.052	22
SP3	1.030	603	1.001	66

Table 9 Improvements as a function of λ as obtained by SP3, Class 1 instances

SP3	Initial solution obtained	$\lambda = 5$	$\lambda = 10$	$\lambda = 20$	$\lambda = 50$
Value	52728	51163	50907	50293	49808
Value/Best	1.087	1.056	1.051	1.042	1.030
Seconds (cumulative)	197	242	319	455	800

Results presented in Tables 2 and 3 indicate that IP1 is better but slower than IP2. On all classes and sizes of instances, IP1 yields a lower deviation from the best known or optimal solution value than IP2 does. On the twenty class 2 instances, the solution values produced by IP1 are within 0.5% of the optimum, and the optimum is reached twelve times of twenty. Moreover, our preliminary tests have shown that IP1 even with $\lambda = 5$ typically performs better than IP2 with $\lambda = 50$.

The two AP heuristics are compared in Tables 4 and 5. For most instances, heuristic AP2 is better than AP1, but there are several cases where this is not true. However, AP2 is considerably more time consuming than AP1. Overall, neither AP1 nor AP2 fares as well as IP1.

The solution perturbation scheme is really the best of all three, except perhaps for SP2 which did not always perform very well. Heuristics SP3, which is obtained from SP1 by introducting a cross-over operation, tends to be better than the latter approach. On class 1 instances, it takes about twice as long as SP1. On class 2 instances, SP1 and SP3 have about the same computing time since

SP3 converges quickly towards the optimum and thus performs fewer iterations. On these instances, SP3 identifies the optimum sixteen times out of twenty. We have summarized in Table 8 the behavior of the seven perturbation schemes over all instances.

A final comment concerns the effect of applying a perturbation scheme to a locally optimal solution. This is best illustrated by studying the effect of λ (number of consecutive applications of the perturbation and postoptimization steps without improvement) when SP3 is applied with the recommend selection of parameters. Our analysis is presented in Table 9 for class 1 instances. The first column of this table corresponds to the locally optimal solution obtained at the end of Step 1. It has a Value/Best ratio equal to 1.087. Executing SP3 with $\lambda = 5$ brings this ratio down to 1.056 and requires approximately 45 seconds. If one is willing to invest more time, this ratio can be brought down even further, but the marginal benefit eventually levels off.

4. Conclusion

We have conducted a computational comparison of seven perturbation heuristics for the PDTSP. This study makes two distinct contributions. It first provides a classification and performance analysis of several types of perturbation schemes applicable to a wide range of combinatorial optimization problems. These provide an efficient means of escaping from local optima in iterative search algorithms. Our experiments show that applying solution perturbation can be quite powerful. A second contribution of this study is the development of a highly efficient heuristic for the PDTSP. On class 1 instances, the best of our perturbation schemes improves the previous results of Renaud, Boctor and Ouenniche (2000) by approximately 4%, and on class 2 instances it consistently yields optimal or near-optimal solutions.

Acknowledgements

This research was partly supported by the Canadian Natural Sciences and Engineerings Research Council under grants OGP0172633, OGP0036509 and OGP0039682. This support is gratefully acknowledged.

References

Boctor F.F., Discrete optimization and multi-neighbourhood local improvement heuristics, Working paper 93–35, Faculté des sciences de l'administration, Université Laval, 1993.

Casco D.O., B.L. Golden and E.A. Wasil, Vehicle routing with backhauls: models, algorithms and case studies, 1988. In B.L. Golden and A.A. Assad (eds.), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam, pp. 121–147.

Charon I., O. Hudry, The noising method: a new method for combinatorial optimization, *Operations Research Letters*, 1993, **14**, 133–137.

Codenotti B., Manzini, G., Margara L. and G. Resta, Perturbation: An efficient technique for the solution of very large instances of Euclidean TSP, *INFORMS Journal on Computing*, 1996, **8**, 125–133.

Desrosiers J., Dumas Y. and F. Soumis, A dynamic programming solution to the large-scale single-vehicle dial-a-ride problem with time windows, *American Journal of Mathematics and Management Science*, 1986, **6**, 301–325.

Dowsland K., Genetic algorithms – a tool for OR?, *Journal of the Operational Research Society*, 1996, **47**, 550–561.

Dumas Y., J. Desrosiers, and F. Soumis, The pick-up and delivery problem with time windows, *European Journal of Operational Research*, 1991, **54**, 7–22.

Eglese R.W., Simulated annealing: A tool for operational research, *European Journal of Operational Research*, 1990, **46**, 271–281.

Fleurent C. and J. Ferland, A Hybrid Genetic Algorithms in Combinatorial Optimization, *Recherche opérationnelle*, 1996, **30**, 373–398.

Gendreau M., Hertz A. and G. Laporte, An approximation algorithm for the traveling salesman problem with backhauls, *Operations Research*, 1997, **45**, 639–641.

Gendreau M., Hertz A. and G. Laporte, The traveling salesman problem with backhauls, *Computers & Operations Research*, 1996, **23**, 501–508.

Glover F. and M. Laguna, *Tabu Search*, 1997, Kluwer, Boston.

Goetschalckx M. and C. Jacobs-Blecha, The vehicle routing problem with backhauls, *European Journal of Operational Research*, 1989, **42**, 39–51.

Healy P. and R. Moll, A new extension of local search applied to the dial-a-ride problem, *European Journal of Operational Research*, 1995, **83**, 83–104.

Jaw J., Odoni A.R., Psaraftis H.N. and N.H.M. Wilson, A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows, *Transportation Research B*, 1986, **20**, 243–257.

Kalantari B., Hill A.V. and S.R. Arora, An algorithm for the traveling salesman problem with pickup and delivery customers, 1985, *European Journal of Operational Research*, **22**, 377–386.

Lin S. Computer solutions to the traveling salesman problem, *Bell Systems Technical Journal*, 1965, **44**, 2245–2269.

Mingozzi A., Giorgi S. and R. Baldacci. An exact method for the vehicle routing problem with backhaul, *Transportation Science*, 1999, **33**, 315–329.

Mladenovic N. and P. Hansen, Variable neighbourhood search, *Computers & Operations Research*, 1997, **24**, 1097–1100.

Padberg M.W. and G. Rinaldi, A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Review*, 1991, **33**, 60–100.

Psaraftis H., A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem, *Transportation Science*, 1980, **14**, 130–154.

Psaraftis H., Analysis of an $O(N^2)$ heuristic for the single vehicle many-to-many euclidean dial-a-ride problem, *Transportation Research*, 1983, **17B**, 133–145.

Psaraftis H.N., Scheduling large-scale advance-request dial-a-ride systems, *American Journal of Mathematical and Management Sciences*, 1986, **6**, 327–367.

Reinelt G., TSPLIB-A traveling salesman problem library, *ORSA Journal on Computing* 1991, **3**, 376–384.

Renaud J., Boctor F.F. and G. Laporte, A fast composite heuristic for the symmetric traveling salesman problem, *INFORMS Journal on Computing*, 1996, **8**, 134–143.

Renaud J., Boctor F.F. and I. Ouenniche, A heuristic for the pickup and delivery traveling salesman problem, *Computers & Operations Research*, 2000. Forthcoming.

Savelsbergh M.W.P., An efficient implementation of local search algorithms for constrained routing problems, *European Journal of Operational Research* 1990, **47**, 75–85.

Savelsbergh M.W.P. and M. Sol, The general pickup and delivery problem, *Transportation Science*, 1995, **29**, 17–29.

Sexton T.R. and L.D. Bodin, Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling, *Transportation Science*, 1985a, **19**, 378–410.

Sexton T.R. and L.D. Bodin, Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing, *Transportation Science*, 1985b, **19** 411–435.

Sexton T.R. and Y. Choi, Pickup and delivery of partial loads with time windows, *American Journal of Mathematical and Management Sciences*, 1986, **6**, 369–398.

Storer R., Wu, S.D. and R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, *Management Science*, 1992, **38**, 1495–1509.

Van der Bruggen, L.J.J., Lenstra J.K. and P.C. Schuur, Variable-depth search for the single-vehicle pickup and delivery problem with time windows, *Transportation Science*, 1993, **27**, 298–311.